A Lightweight Mitigation Technique for Resource-constrained Devices Executing DNN Inference Models under Neutron Radiation

Jonas Gava, *Student Member, IEEE,* Alex Hanneman, *Student Member, IEEE,* Geancarlo Abich, *Student Member, IEEE,* Rafael Garibotti, *Senior Member, IEEE,* Sergio Cuenca-Asensi, Rodrigo Possamai Bastos, *Member, IEEE,* Ricardo Reis, *Life Senior Member, IEEE,* and Luciano Ost *Senior Member, IEEE*

***Abstract*—Deep neural network (DNN) models are being deployed in safety-critical embedded devices for object identification, recognition, and even trajectory prediction. Optimised versions of such models, in particular the convolutional ones, are becoming increasingly common in resource-constrained edge-computing devices (e.g., sensors, drones), which typically rely on reduced memory footprint, low power budget and low-performance microprocessors. DNN models are prone to radiation-induced soft errors, and tackling their occurrence in resource-constrained devices is a mandatory and substantial challenge. While traditional replication-based soft error mitigation techniques will likely account for a reasonable performance penalty, hardware solutions are even more costly. To undertake this almost contradictory challenge, this work evaluates the efficiency of a lightweight software-based mitigation technique, called Register Allocation Technique (RAT), when applied to a convolutional neural network (CNN) model running on two commercial Arm microprocessors (i.e., Cortex-M4 and M7) under the effects of neutron radiation. Gathered results obtained from two neutron radiation campaigns suggest that RAT can reduce the number of critical faults in the CNN model running on both Arm Cortex-M microprocessors. Results also suggest that the SDC FIT rate of the RAT-hardened CNN model can be reduced in up to 83% with a runtime overhead of 32%.**

***Index Terms*—Neutron Radiation, DNN and CNN inference models, Resource-constrained Devices, Arm Cortex-M.**

I. Introduction

DEEP neural network (DNN) models notoriously demand high computational power and resources (e.g., memory footprint), which imposes fundamental challenges on their deployment in resource-constrained edge devices that integrate low-power microprocessors and sensors. The availability of more efficient edge-computing devices and specialised libraries/APIs has enabled a wide variety of DNN-based solutions for systems and applications across several embedded computing domains. The growing adoption of DNN models in safety-critical embedded systems (e.g., autonomous vehicles and drones), increases the demand for more efficient, safe and reliable models. In this regard, authors have investigated different fault-aware training approaches [1][2][3] aiming to improve the robustness of neural network models, for instance, by reducing the occurrence of memory faults. Although efficient such kind of approach may be retraining and/or dataset dependable [1]. Datasets are not always available and the retraining cost may not be affordable to complex accelerators or every single business case [4]. More traditional hardening approaches (e.g., triple modular redundancy - TMR) have also been either adapted for DNN solutions implemented in FPGAs [5][6][7] or applied to DNN models running in specialised accelerators [8][9]. However, the lower resource availability of edge-computing devices makes traditional redundancy mitigation approaches unsuitable for tackling the occurrence of radiation-induced soft errors, given their likely impact on the system's performance and response time.

Based on the above, we advocate that to achieve high safety and reliability standards, resource-constrained edge devices must be underpinned by efficient and lightweight mechanisms able to reduce their vulnerability to radiation-induced soft errors with minimal power and performance overhead. Further, the complex nature of DNN models calls for more efficient approaches that enable automatic and bespoke code hardening.

This paper *contributes* by investigating the efficiency of the Register Allocation Technique (RAT) [10] when applied to a convolutional neural network (CNN) model running on an Arm Cortex-M4 and M7 microprocessors under the effects of neutron radiation. RAT is a compiler-based technique that does not involve code redundancy, but rather restricts the number of available registers used to execute specific functions, thus reducing the exposed area. Results show that the promoted technique can reduce the number of critical faults and the SDC FIT rate of the CNN inference model with a runtime overhead of 32%.

The rest of this paper is organised as follows. Section II presents related works in the assessment and mitigation of soft errors in ML models operating under radiation effects. Section III describes the adopted hardening technique. In

This work was partially funded by: CAPES; CNPq (317087/2021-5); FAPERGS (22/2551-0000570-5); UK EPSRC (EP/R513088/1); MultiRad (PAI project funded by Région Auvergne-Rhône-Alpes); IRT Nanoelec (ANR-10-AIRT-05 project funded by French PIA); UGA/LPSC/GENESIS platform; and PID2019-106455GB-C22 (funded by the Spanish Ministry of Science and Innovation).

J. Gava, G. Abich and R. Reis are with PGMicro, UFRGS, Brazil (e-mail: {jfgava, gabich, reis}@inf.ufrgs.br).

A. Hanneman and L. Ost are with Wolfson School, Loughborough University, UK (e-mail: {a.d.hanneman, l.ost}@lboro.ac.uk).

R. Garibotti is with School of Technology, PUCRS, Brazil (e-mail: rafael.garibotti@pucrs.br).

S. Cuenca-Asensi is with Departamento de Tecnología Informática y Computación, Universidad de Alicante, Spain. (e-mail: sergio@dtic.ua.es).

R. Possamai Bastos is with Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, France. (e-mail: rodrigo.bastos@univ-grenoble-alpes.fr).

Corresponding author: Luciano Ost (e-mail: l.ost@lboro.ac.uk).

Manuscript received October 1, 2022.

Section IV the CNN inference model, adopted as case-study, is detailed. Section V details the methodology used perform the radiation tests. Section VI discusses the effectiveness of RAT. Finally, Section VII points out conclusions.

II. Related Work in Machine Learning Soft Error Assessment and Mitigation

The soft error assessment and mitigation literature is abundant, requiring a taxonomy to classify the different approaches. This work considers the definitions from [20] for fault, error, and failure. A fault is an event that may cause the internal state of the system to change, e.g., a radiation particle strike. When a fault affects the system's internal state, it becomes an error. If the error causes a deviation of at least one of the system's external states, it is considered a failure. To achieve compliance with safety and reliability standard requirements, it is of utmost importance to provide systems with appropriate mechanisms to tackle systematic or transient faults, also known as soft errors or Single Event Upsets (SEUs). While the former originates from hardware and software design defects, soft errors are those caused by alpha particles or atmospheric neutrons [21]. The occurrence of soft errors can be tackled both in hardware and software. While hardware approaches lead to the area and power overhead, software techniques are generally implemented on a per-application basis that usually incurs performance penalties.

Due to the ever-increasing trend towards having ML models embedded in edge safety-critical systems, researchers have started to investigate the impact of radiation-induced soft errors on the reliability of underlying models considering simulation (e.g., [22][23][24][25][26][27][28][29]), emulation (e.g., [6][14][30]) and radiation tests (see Table I). While high-level simulation and emulation approaches are highly useful to conduct early soft error assessment and eliminate not suitable options, final systems' configurations must be evaluated through radiation tests. Table I summarises the works that investigate the impact of radiation-induced soft errors on the reliability of ML algorithms and trained models under radiation tests, highlighting those approaches that employ a mitigation technique.

Except for our pioneering works ([14][19]), this is the only work employing resource-constrained devices in the experiments. The remaining works consider FPGA implementations of ML algorithms [5][12][6][15][17][7] or their execution on generic graphics processing units (GPUs) [11], and ML specialised accelerators [13][16][18]. All works, with exception of [5], adopted neutron irradiation for their experiments.

On the soft error mitigation side, in addition to this paper only three works have considered hardened inference models in their radiation experiments [6][16][11]. In this context, Libano *et al.* [6] promote the selective hardening approach, where the most sensitive NN and CNN layers are fully or partially triplicated in the FPGA. Blower *et al.* [16] focus on an artificial neural network (ANN) running on the NeuroShield accelerator. This work proposed a bespoke TMR-based solution dedicated to the target accelerator, designed based on experimental observations. Results show that 96% of critical errors were reduced at no performance cost. However, the hardening solution demands error propagation analysis, in-depth dataset knowledge and modification, and model retraining. Further, the efficiency of this hardening method depends on the number of classification classes supported by the NN model. In turn, Santos *et al.* [11] focus on a more generic Graphics Processing Unit (GPU), which is widely used in high-computational systems. In this work, the Algorithm-Based Fault Tolerance (ABFT) technique has been applied due to its efficient to detect soft errors in dense linear algebra operations including matrix multiplication, which form a significant part of the CNNs operation. Although ABFT brings less performance overhead w.r.t. replication-like approaches (e.g., TMR), resulting overhead might still lead to high response times - a critical consideration for resource-constrained devices.

Unlike the above mitigation techniques, RAT is a compiler-based hardening solution; thus, neither dataset modification nor model retraining is needed. Instead, RAT enables automatic hardening of the most critical function(s) or layer(s) of generic or DNN-based applications, considering the possibility of manual configurations to guide bespoke hardening tuning for resource-constraint devices.

This work distinguishes from the previous works in two main aspects:

– First, this is the first work to assess the soft error susceptibility of a CNN inference model considering two resource-bound microprocessors, presented in commercial smart sensor systems [31] and drone platforms [32][33], under several radiation exposure hours.
– Second, a comprehensive evaluation of the efficiency of RAT on a CNN inference model executing on Arm Cortex-M microprocessors under radiation effects is presented for the first time.

III. Adopted Soft Error Mitigation Technique

This work adopts the lightweight technique, called Register Allocation Technique (RAT) [10], to mitigate the occurrence of soft errors in a 7-layer CNN inference model. The RAT technique does not involve code redundancy and is an architecture-independent approach, which restricts the number of available registers for specific functions aiming to minimise the number of vulnerable registers. RAT is implemented on the backend of the LLVM compiler, and can be fine-tuned to achieve the highest relative trade-off in terms of performance and soft error reliability. Thus, the software engineer can set the registers' pool parameter, aiming to restrict the application's register allocation considering their availability in the target architecture, and the minimum number of registers demanded by each application. Furthermore, the software engineer can also set a list of application functions (e.g., the critical ones) that must be hardened. Functions are limited to those available in the program's source code; that is, functions from external libraries do not suffer any effect as they are already in the machine code format.

RAT has been integrated into SOFIA [34], a fully automated framework that supports fast and early soft error assessment,

TABLE I
RELATED WORKS IN THE ASSESSMENT AND MITIGATION OF SOFT ERRORS IN MACHINE LEARNING MODELS UNDER RADIATION EFFECTS.

Work	Dataset	ML Algorithm/Model	Assessment Aproach	Target Device	Radiation Fluence	Mitigation
Libano *et al.* [5] (2018)	Boston Housing, Iris Flower	2-layer NN	Emulation, Heavy-ion irradiation	Xilinx Zynq-7000	$1.14 \times 10^7\ \mathrm{i/cm^2}$	—
Santos *et al.* [11] (2018)	PASCAL VOC, Caltech Pedestrian	YOLO, Faster R-CNN, ResNet	Simulation, Neutron irradiation	NVIDIA GPU	$7.2 \times 10^{11}\ \mathrm{n/cm^2}$ *	ABFT
Trindade *et al.* [12] (2019)	Fault Detection	Support Vector Machine (SVM)	Neutron irradiation	Xilinx Zynq-7000	$1.944 \times 10^{10}\ \mathrm{n/cm^2}$	—
Libano *et al.* [6] (2019)	Iris Flower, MNIST	2-layer NN, 7-layer CNN	Emulation, Neutron irradiation	Xilinx Zynq-7000, Zynq Ultrascale+	$1.6 \times 10^{11}\ \mathrm{n/cm^2}$, $3.0 \times 10^{11}\ \mathrm{n/cm^2}$	Partial TMR, Full TMR
Brewer *et al.* [13] (2020)	MNIST	SNN	Emulation, Neutron irradiation	TrueNorth Neurosynaptic	$2 \times 10^7\ \mathrm{p/cm^2}$ - $5.6 \times 10^7\ \mathrm{p/cm^2}$	—
Trindade *et al.* [14] (2020)	Iris Flower	ANN, SVM	Emulation, Neutron irradiation	Arm Cortex-M4	$2.916 \times 10^9\ \mathrm{n/cm^2}$	—
Luza *et al.* [15] (2020)	MNIST	LeNet-5 CNN	Neutron irradiation	Xilinx Zynq-7000	**	—
Blower *et al.* [16] (2021)	MNIST	ANN	Neutron irradiation	NeuroShield	$1.5 \times 10^8\ \mathrm{n/cm^2}$	Partial TMR
Libando *et al.* [17] (2021)	MNIST	CNN	Neutron irradiation	FPGA	$3.44 \times 10^{11}\ \mathrm{n/cm^2}$	—
Wang *et al.* [7] (2021)	CIFAR-10	CNN	Neutron irradiation	FPGA	$2.91 \times 10^8\ \mathrm{n/cm^2}$, $2.67 \times 10^8\ \mathrm{n/cm^2}$	—
Rech *et al.* [18] (2022)	COCO, ILSVRC, Oxford-IIIT Pet	Inception v4, ResNet-50, SSDLite MobileNet V2, SSDLite MobileDet	Neutron irradiation	Google Coral USB TPU	$3.41 \times 10^{12}\ \mathrm{n/cm^2}$	—
Bastos *et al.* [19] (2022)	Iris Flower	ANN, SVM, RF	Neutron irradiation	Arm Cortex-M4	$1.9\text{–}2.4\times10^9\mathrm{n/cm^2}$, $1.0\text{–}1.2\times10^7\mathrm{n/cm^2}$	—
This work (2022)	**CIFAR-10**	**CMSIS CNN**	**Neutron irradiation**	**Arm Cortex-M4, Arm Cortex-M7**	$1.98 \times 10^{10}\ \mathrm{n/cm^2}$ $3.18 \times 10^{10}\ \mathrm{n/cm^2}$	**RAT**

*Approximate lower bound of total fluence per device
**Insufficient information to calculate

and includes other mitigation techniques such as partial and full TMR. SOFIA provides orders of magnitude speedup while preserving the soft error analysis accuracy (i.e., mismatch below to 10%) w.r.t. the RTL level [35]. SOFIA is a complementary reliability assessment approach to radiation testing and does not introduce any overhead on the radiation-exposed hardened or unhardened application. The main steps to apply RAT automatically using SOFIA are shown below:

1) *Software stack setup*, which includes the application, operating system, and driver configuration;
2) *Compilation using LLVM* with the correct target processor parameters;
3) *Dynamic analysis*. In this step, the application is simulated and essential information is extracted (i.e., processor register file utilisation and critical function);
4) *Static analysis*, which investigates the application object code extracting information about the functions and registers usage;
5) Finally, a *new compilation* is performed, taking into account the critical function and the register pool previously set. The underlying compilation uses a modified version of the LLVM Fast Register Allocator, which considers arguments (i.e., restrictions) that are passed to LLVM Static Compiler (LLC) through a command line.

Note that the RAT technique does not work properly on modern out-of-order processors that use register renaming. However, RAT can increase the soft error reliability of applications running on resource-constrained devices that employ simple in-order and low-power microprocessors, whereas its advantages have been fully explored and demonstrated, in simulation, considering the MobileNet CNN on ImageNet dataset [25]. This is the first work to investigate the RAT effectiveness when applied to a CNN inference model running on low-power microprocessors under neutron radiation.

IV. CASE-STUDY

This case study comprises a 7-layer CNN based on the CMSIS-NN library [36], which is deployed on two Arm Cortex-M microprocessors. Unlike traditional libraries and NN models, CMSIS-NN uses low-precision fixed-point representation to set the inference model. This feature brings performance advantages to resource-constrained devices, which may not have a dedicated Floating-Point Unit (FPU) or large memory footprints available. Furthermore, as high-precision representation is generally not required during inference, this approach avoids the need for floating-point de-quantization between layers and reduces the memory footprint.

The adopted CNN inference model comprises convolution layers interspersed by non-linear activation layers, pooling layers, and a fully-connected layer at the end of CNN. The convolution layer implements a partial image-to-column data transformation considering the Height-Width-Channel (HWC) format. Rectified Linear activation Unit (ReLU) layer loops through all elements replacing negative values with zeros.

Then, pooling layers reduce the number of parameters, the feature dimensions, and the network computations. Finally, the fully-connected layer has connections to all activations in the previous layer to define the output classification probabilities. The RAT mitigation technique was applied manually to the first convolutional layer, which is the most executed one. In this sense, the register pool has been restricted to R0-R4, in addition to the special registers (SP, PC) that cannot be altered during the allocation phase.

The CNN model is trained with the CIFAR-10 dataset [37], consisting of 60,000 32x32 colour images divided into ten output classes. The choice for this generalist dataset, which contains automobiles, planes, ships, trucks, and some animals, is due to its usability in the most diverse applications under resource-constrained devices available today, such as monitoring boats or livestock [38].

V. Radiation Test Methodology

This Section describes the methodology used to collect and show the results obtained with a 14MeV neutron generator, which has been used to test the effectiveness of RAT when applied to a CNN inference model running on two resource-constrained test boards under neutron radiation.

A. Radiation Test Set-up

Two 14-MeV neutron radiation test campaigns were performed at the LPSC (Grenoble, France), the first in November 2021 and the second in July 2022. The two radiation test campaigns used the GENEPI2 (GEnerator of NEutrons Pulsed Intense) neutron source, a neutron generator that delivers 14-MeV neutron beam with a maximum flux that exceeds the natural 14-MeV neutron flux at 40,000 ft by a factor of 10^{10}. Note that a total fluence $> 5.4 \times 10^{10}$ neutron/cm^2 was chosen to achieve statistical significance. In this regards, the average flux during the experiment in November 2021 was 9.2×10^6 neutron/cm^2/s, while the average flux during the radiation tests in July 2022 was 7.4×10^6 neutron/cm^2/s.

Figure 1 (left photo) illustrates the set-up assembled at the LPSC. The STM32-L476RGT6U board [39] and the STM32-F767ZIT6 board [40] were selected as the target devices. They were chosen to cover Arm Cortex-M microprocessors with single instruction and multiple data (SIMD) capabilities, i.e., the Arm Cortex-M4 board was evaluated in November 2021 (Figure 1 - centre photo) and the Arm Cortex-M7 board was evaluated in July 2022 (Figure 1 - right photo). Figure 1 also shows that the whole system (CPU, memory, communication peripherals) is under the beam and that the STM32 boards were placed in the first boards row of the neutron generator.

Table II shows the configuration of the test boards, which have been set to their maximum clock frequency and with all caches enabled. We refer to the test boards as a device under test (DUT) onward. In regard to the application configuration, the CNN model was compiled using Clang/LLVM 6.0.1 with O2 optimisation level. Furthermore, the two experiments use the same tiny 32x32 dog image (3kB) initialised in the RAM, as input of the CNN model. The output is an integer vector of 10 elements, which indicates the probability for the distinct classifications. The majority of the CNN parameters (e.g., weights and bias) are stored in Flash memory (about 100kB), and about 50kB of RAM is reserved for storing partial results during the CNN computation.

TABLE II
Test Board Configuration.

Microprocessor	Arm Cortex-M4	Arm Cortex-M7
MCU (STM32-)	L476RGT6U	F767ZIT6
Clock (MHz)	80	216
Flash (kB)	1,024	2,048
RAM (kB)	128	512
I Cache (kB)	-	16
D Cache (kB)	-	16
Cache	-	I & D
Firmware	L4 1.17.0	F7 1.16.1

B. Radiation Test Flow

Figure 2 shows the test flow schematic. First, the flux was calibrated remotely to fit a proper operation of the DUTs, which is connected to a control computer (CC) outside the radioactive chamber through a USB cable. Then, the Universal Asynchronous Receiver Transmitter (UART)-based communication between the DUT and the CC is verified via checkers (i.e., checksum) to isolate radiation-induced failures in the UART peripheral and the data communication channel between the DUT and the CC. The communication time takes up to 20 ms in total for each run, which corresponds to 2-4% in relation to the CNN computation time. The checksum computation takes less than 1 ms.

The steps to run the radiation tests are shown below:

1) Board programming using the Open On-Chip Debugger (openocd);
2) Synchronise both DUT and CC devices before the algorithm main function execution, i.e., send a message from CC to DUT and awaits the correct response. If the DUT takes more than 5 seconds to respond, the board is then reprogrammed;
3) Check the input checksum. The checksum is calculated in the DUT and sent to the CC for checking against the golden reference. If there is a mismatch in the input data, the board is reprogrammed. If the reprogramming fails, the relay is actioned, and a power cycle is done;
4) Algorithm main function execution;
5) Synchronise DUT and CC after the algorithm computation, i.e., send a message from CC to DUT and awaits the correct response;
6) Send output and output checksum from DUT to CC. The checksum is calculated in DUT and sent to CC. If there is a mismatch in the output, the procedure is the same as for the input data.

C. Adopted Fault Classification and Reliability Metrics

The most commonly used classification to describe a failure in a device exposed to radiation considers silent data corruption (SDC) and detectable unrecoverable error (DUE) as main classes. However, to better understand the impact of soft errors

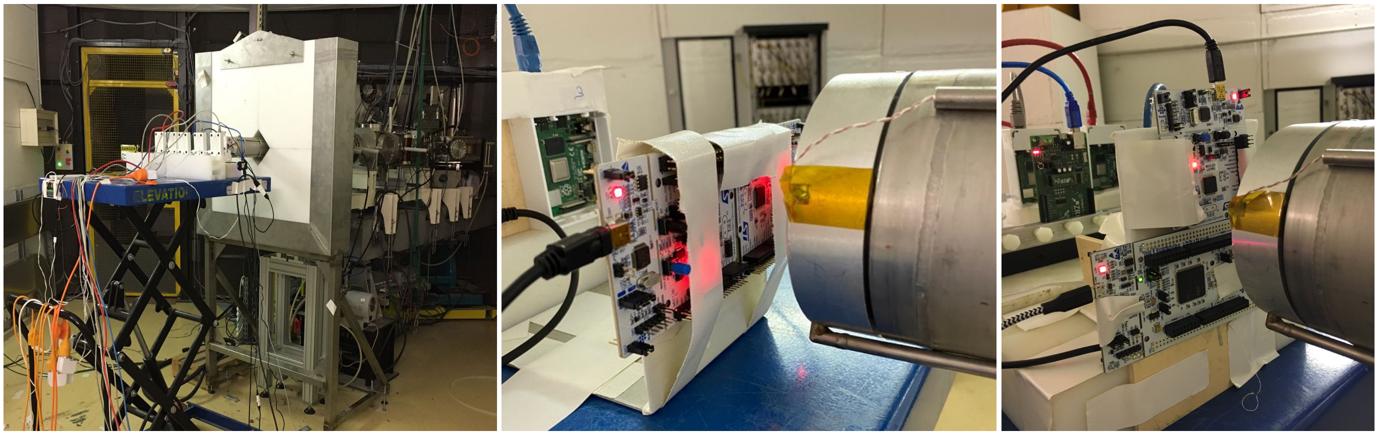

Fig. 1. DUTs mounted at the LPSC facility (left photo), highlighting the use of Arm Cortex-M4 board in November 2021 (centre photo) and Arm Cortex-M7 board in July 2022 (right photo).

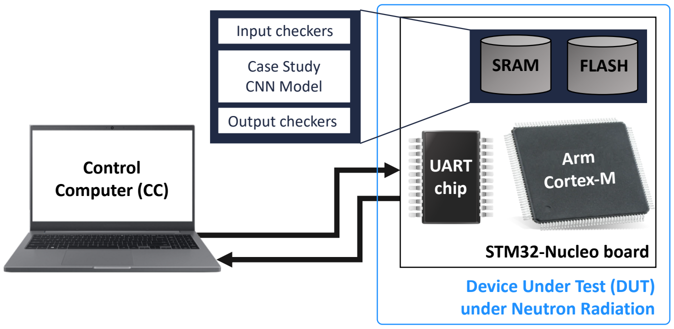

Fig. 2. The radiation test flow schematic, highlighting the communication between the control computer and the system under test with its check system.

on CNN models, this work classifies the radiation results into three different types. *Tolerable* is when there is a mismatch between the CNN model result and its golden reference. However, they present a top-ranked classification equals to the fault-free execution. On the other hand, *critical* is when there is a mismatch between the CNN model result and its golden reference, and the resulting classification is incorrect. Note that a typical SDC comprises these two fault types, i.e., tolerable and critical. The last classification is *crash*. It occurs when the CNN algorithm suffers from abnormal termination or application hang. Furthermore, if the communication between CC and DUT is lost during the CNN execution, it indicates that radiation effects have upset the DUT. In this situation, the board must be restarted and classified as a *crash*. Note that this paper uses the above fault classification to measure the $Tolerable_{SDC}$, $Critical_{SDC}$ and $Crash$ events for the hardened and unhardened execution of the CNN.

The following metrics are used to compare the reliability of both hardened and unhardened CNN model executions. The Failure in Time (*FIT*) metric shows how many failures occur in a billion hours. It depends on both the device sensitivity and the particle flux to which it will be exposed. The specification that defines standard requirements and procedures for terrestrial soft-error-rate testing of integrated circuits and reporting of results (JEDEC) suggests to uses 13 n/cm^2/h as flux using NYC as a reference [41]. Furthermore, the cross-section of the device for a singular test was calculated using Equation (1), where σ is the per-bit cross-section, N_e is the number of errors, and ϕ is the neutron fluence.

$$\sigma = \frac{N_e}{\phi} \tag{1}$$

This work also uses the Mean Work to Failure (MWTF), which captures the average work the CNN can perform between failures, to assess the the trade-off between reliability improvement and runtime overhead. Equation (2) shows how it is calculated.

$$MWTF = \frac{1}{(\sigma \times flux \times execution\ time)} \tag{2}$$

The execution time refers to the rate at which a CNN can be invoked, calculated from the cycles needed to run the CNN inference model. This was measured using the processor's hardware performance counters and considering both Cortex-M4 and M7 running at their maximum clock speeds so the cycle count can be considered a worst case for both microprocessors.

VI. Radiation Results

This Section explores the soft error reliability of a CNN inference model under neutron radiation. In this sense, Section VI-A provides an overview of the radiation results, showing reliability trends from the hardening approach. Then, each campaign (i.e., Arm Cortex-M4 and M7 board) is analysed separately. Section VI-B details the results obtained with the Arm Cortex-M4 boards in the radiation test campaign performed in November 2021. Next, Section VI-C exploits the CNN inference model soft error reliability considering the Arm Cortex-M7 microprocessor - campaign held in July 2022.

A. General Analysis of Radiation Results

Table III summarises the neutron radiation test results. For the Arm Cortex-M4 microprocessor, a total of 4636 runs were performed for the unhardened version and 4859 runs for the RAT-hardened version. Note that the unprotected CNN model

This article has been accepted for publication in IEEE Transactions on Nuclear Science. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNS.2023.3262448

TABLE III
SUMMARY OF THE NEUTRON RADIATION TEST RESULTS FOR THE TWO RESOURCE-CONSTRAINED TEST BOARDS (ARM CORTEX-M4 IN NOVEMBER 2021 AND ARM CORTEX-M7 IN JULY 2022).

Microprocessor	Case-Study Scenarios	Runs	Runtime [ms]	Effective Fluence [10^{10} neutrons/cm^2]	Events (FIT [Failures/10^9 h])			MWTF (σ [10^{-10} cm^2])		Memory Usage [kB]	
					Tolerable	Critical	Crash	Critical	Crash	RAM	Flash
Arm Cortex-M4	Unhardened	4636	538	2.29	103 (58.17)	6 (3.39)	42 (23.72)	772 (2.6)	110 (18.3)	48.91	93.04
	RAT	4859	712	3.18	22 (8.96)	4 (1.63)	50 (20.36)	1214 (1.2)	97 (15)	48.91	96.69
Arm Cortex-M7	Unhardened	6231	430	1.98	12 (7.8)	6 (3.9)	28 (18.25)	1038 (3.02)	223 (14)	51.53	137.57
	RAT	7500	570	3.16	24 (9.8)	2 (0.8)	19 (7.77)	3750 (0.63)	395 (5.99)	51.53	141.47

runtime is around 538 ms, while the RAT version incur a performance penalty of 32%. However, the results demonstrate that improved soft error reliability offsets this performance overhead. For the Arm Cortex-M7 microprocessor, the underlying results were obtained from 7500 RAT-hardened runs and 6231 runs of the unhardened version. Similar to the Arm Cortex-M4 test, the execution time of the RAT-hardened CNN (i.e., 570 ms) was 32% longer than the unhardened CNN model (i.e., 430 ms).

However, the situation is reversed for the soft error reliability results, which show that the RAT-hardened version presents a 3.1× improvement for $MWTF_{critical}$ and 1.77× for $MWTF_{crash}$. Regarding the cross-section, there is an 80% reduction in $\sigma_{critical}$ and 58% in σ_{crash} when using the RAT-hardened version. Note that for calculating the MWTF metric, the total fluence used for the unhardened version was 1.98×10^{10} n/cm^2 and 3.16×10^{10} n/cm^2 for the RAT-hardened version.

To make our radiation results comparable with the literature, Figure 3 shows the number of SDCs and the FIT rate considering the two Arm Cortex-M boards without and with protection (RAT). The SDC FIT rate (i.e., FIT_{SDC}) is calculated by summing the FIT rate of both tolerable and critical faults. Note that the margin of error follows the JESD89-A standard [42], which considers the total number of memory elements, the total number of events, the inverse cumulative standard normal distribution function ($\phi^{-1}(\alpha/2)$), and a 95% confidence level.

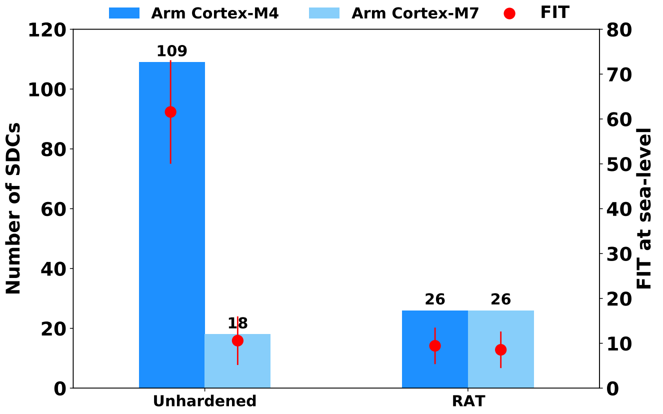

Fig. 3. SDC results for Arm Cortex-M4 and Arm Cortex-M7 boards.

For the Arm Cortex-M4, results show that the FIT_{SDC} reduces by 83.6% when applying RAT. The 83% reduction can be calculated by comparing the first (i.e., 61.56) and the third (i.e., 10.59) red dot values, illustrated in Figure 3, for the Cortex-M4 processor. On the other hand, the Arm Cortex-M7 FIT results are the same (i.e., about 10%) for unhardened and RAT when considering the margin of error. However, although the RAT shows more SDC events, the protected CNN inference model was exposed for a longer period to neutron radiation.

Figure 4 presents the number of executed instructions on the two Arm Cortex-M boards for the unhardened and RAT CNN versions. Results show that the number of executed instructions is similar for both boards. The number on the Arm Cortex-M7 is slightly higher for the unhardened version due to its more significant number of drivers. Unlike, when RAT is applied, the number of executed instructions is closer because the Arm Cortex-M4 has fewer resources than the Arm Cortex-M7 board, requiring more instructions to be executed.

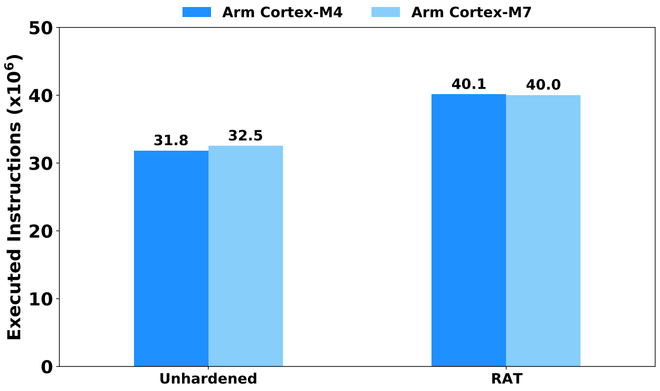

Fig. 4. Executed instructions for Arm Cortex-M4 and Arm Cortex-M7 boards.

After an initial overview of the radiation results, an in-depth analysis is conducted in the next Sections, splitting the SDC into tolerable and critical faults, and analysing the boards separately, in order to draw more robust conclusions about the efficiency of RAT protection.

B. Arm Cortex-M4 Radiation Results

Figure 5 shows the radiation test results considering the CNN unhardened and the hardened version (i.e., RAT protection) following the methodology detailed in Section V. Each bar represents an event type (i.e., *tolerable*, *critical* and *crash*) associated with the left y-axis. The red dots represent the FIT metric for each event associated with the right y-axis. During the radiation test campaigns conducted in November 2021, 227 events related to the CNN inference model running on an Arm Cortex-M4 board were observed.

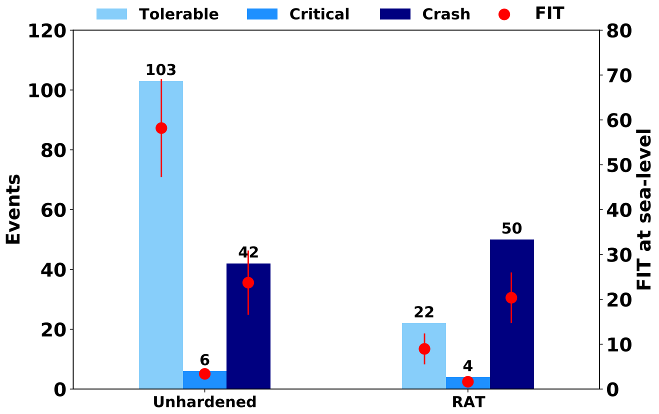

Fig. 5. Radiation-induced failures in each case-study scenario evaluated with the Arm Cortex-M4 board.

Note that communication failures also occurred. However, these were rare and caused by CC problems. On certain occasions, the process used to control the communication with the serial port on the CC hung, changing the port allocated to the DUT. In this scenario, CC kept trying to communicate with the wrong port. This fault is counted as a crash on its first occurrence, while the remaining occasions are discarded until a manual reset is performed.

Figure 5 also shows the FIT rate, which captures the relationship between radiation exposure time and the number of events. Note that the RAT-hardened version has improved the FIT rate on all types of events w.r.t. the unprotected version. As for $\text{FIT}_{\text{tolerable}}$, the RAT version shows a 6.5× improvement. Although it is a significant value, this event does not affect the correct application behaviour. The RAT version also offers a 2.1× improvement for $\text{FIT}_{\text{critical}}$ and 1.1× for the $\text{FIT}_{\text{crash}}$. To explain the $\text{FIT}_{\text{crash}}$ similarity, it is important to understand the category of executed instructions. Figure 6 shows the number of executed instructions divided into three categories: memory access, control, and data processing.

Results show that the RAT version has more instructions devoted to memory access (about 13% - *LOAD/STORE* instructions generated by the compiler). This additional number of *LOAD/STORE* instructions is due to the lack of sufficient registers to hold the variables. Therefore, some variables will have to be moved to the main memory. This directly impacts the data processing proportion, as shown in Figure 6. In this regard, two main reasons may explain the similar $\text{FIT}_{\text{crash}}$ rate: (*i*) the number of incorrect branches from control instructions; and (*ii*) the number of illegal memory accesses from memory instructions (i.e., LOAD/STORE). Note that illegal memory accesses cause a system exception when a fault changes the address to access a prohibited memory space. Works in the literature have demonstrated that SEUs occurring in data or NN parameters (e.g., weights and activation quantizations) stored in memory affect DNN inference models' soft error reliability and accuracy [43].

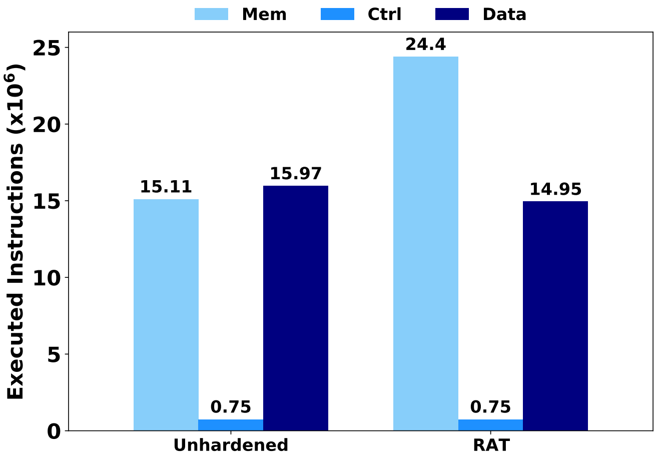

Fig. 6. Dynamic instruction performed on the Arm Cortex-M4 board for the unhardened and RAT versions.

Another important reliability metric is the MWTF, which captures the trade-off between reliability improvement and execution time overhead. For experiments with the Arm Cortex-M4 board, the effective radiation fluence was 2.29×10^{10} [n/cm^2] for the unhardened and 3.18×10^{10} [n/cm^2] for the RAT version. The effective fluence refers only to the application computing time, excluding the time spent managing each run (e.g., communication, checking input and output). Results show that the RAT version provides an $\text{MWTF}_{\text{critical}}$ improvement of 18.8% w.r.t. the unhardened version. For the $\text{MWTF}_{\text{crash}}$, the unhardened version shows an improvement of 50.7%. Note that this is the opposite of the $\text{FIT}_{\text{crash}}$ results, showing that FIT is not the most suitable metric for reliability comparison when considering different versions of the same application.

In regard to the memory utilisation, the protected CNN model has 3.9% higher usage of Flash memory w.r.t. the unhardened version. However, Cellere *et al.* [44] show that Flash memory is quite resilient to neutron radiation, so we can assume that the events did not come from that source. As for RAM usage, both application versions have practically the same memory utilisation, as there is no data replication in the RAT technique.

C. Arm Cortex-M7 Radiation Results

Figure 7 shows the radiation test results considering the CNN unhardened and the hardened version (i.e., RAT protection) running on the Arm Cortex-M7 microprocessor. Each bar represents an event type (*tolerable*, *critical* and *crash*) associated with the left y-axis. The red dots represent the FIT metric for each event associated with the right y-axis.

During the radiation test campaigns conducted in July 2022, 91 events related to the Arm Cortex-M7 board were observed. Figure 7 also shows the relationship between radiation exposure time and the number of events through the FIT rate. On the one hand, the unhardened version shows a 1.25× improvement to $\text{FIT}_{\text{tolerable}}$. On the other hand, the RAT-hardened version offers a 4.78× improvement for $\text{FIT}_{\text{critical}}$ and 2.35× for $\text{FIT}_{\text{crash}}$. These results evidence that the RAT-hardened

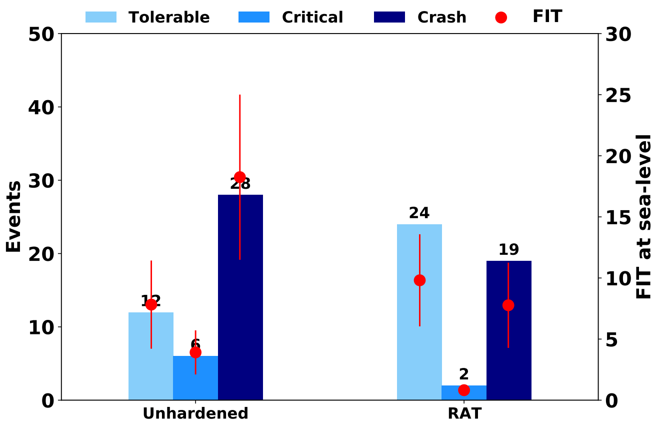

Fig. 7. Radiation-induced failures in each case-study scenario for the Arm Cortex-M7 board.

version improved soft error reliability w.r.t. the unprotected version running on the Arm Cortex-M7 board.

These results suggest that RAT improves the CNN soft error reliability with low memory utilisation and runtime overheads. Observed results also corroborate with those presented by Abich *et al.* [25], which simulated more than 4.5 million fault injections and showed that the RAT technique significantly reduced the occurrence of soft errors of the MobileNet CNN model in the Arm Cortex-M7 microprocessor architecture. Results also show that the MobileNet soft error reliability varies with the precision bitwidth of its convolutional layers and that the reduction of weights and activations precision bitwidth increases the fault-masking capability by up to 10%, thus reducing the MobileNet CNN susceptibility to the occurrence of soft errors. Even under these conditions, the RAT technique improved the MWTF by up to 4.7× compared to the unhardened version.

Some trends can be observed when comparing the neutron radiation test results for the two Arm microprocessor models. First, Figure 5 and Figure 7 show that RAT significantly reduces critical faults, which are the most important events for a system's reliability. While for the Arm Cortex-M4 it is not possible to draw a definitive conclusion due to the margin of error, the same does not occur for the Arm Cortex-M7. However, there is no direct relationship between both microprocessors regarding tolerable faults and crashes.

The results only suggest that the CNN running on the Arm Cortex-M7 board is more resilient to crashes. Second, it is possible to observe that the number and type of instructions executed are very similar when comparing Figure 6 and Figure 8. In fact, the most significant changes between the two Cortex-M microprocessor models are in the pipeline and operation frequency since the architecture is the same. Moreover, the same memory usage pattern is observed for both boards. However, the Arm Cortex-M7 has a significant increase (about 40%) in its Flash memory usage, mainly due to new board drivers.

VII. Conclusion

This work establishes the viability of using a lightweight mitigation technique as an effective alternative to enhance the soft error reliability a CNN inference model running on two resource-constrained devices under neutron radiation. Radiation results suggest that RAT improves the CNN soft error reliability against critical faults with a slight increase in memory utilisation and execution time for both tested STM32 boards. Furthermore, a significant reduction in crashes was observed even with a higher number of runs on the STM32 board using the Arm Cortex-M7, which presented a higher resilient to radiation-induced errors w.r.t. the STM32 Arm Cortex-M4 board.

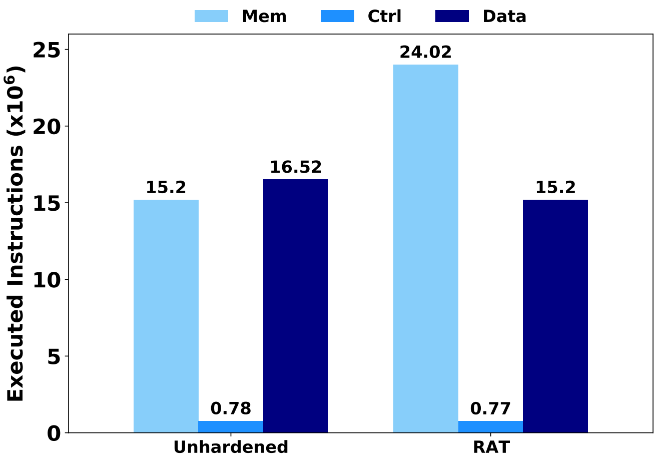

Fig. 8. Dynamic instruction executed for the unhardened and RAT versions running on Arm Cortex-M7.

Future works include to apply RAT to other CNN models, such as AlexNet network, taking into account other Arm and RISC-V microprocessor architectures. We also intend to combine and compare RAT with other mitigation techniques developed in a lower code-level, aiming to surpass actual limitations and consider architectures with more resources available.

References

[1] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. Sathe, "MATIC: Learning around errors for efficient low-voltage neural network accelerators," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 1–6.

[2] U. Zahid, G. Gambardella, N. J. Fraser, M. Blott, and K. Vissers, "FAT: Training Neural Networks for Reliable Inference Under Hardware Faults," in *IEEE International Test Conference (ITC)*, 2020, pp. 236–245.

[3] T. Spyrou, S. A. El-Sayed, E. Afacan, L. A. Camuñas-Mesa, B. Linares-Barranco, and H.-G. Stratigopoulos, "Neuron Fault Tolerance in Spiking Neural Networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 743–748.

[4] L.-H. Hoang, M. A. Hanif, and M. Shafique, "FT-ClipAct: Resilience Analysis of Deep Neural Networks and Improving their Fault Tolerance using Clipped Activation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1241–1246.

[5] F. Libano, P. Rech, L. Tambara, J. Tonfat, and F. Kastensmidt, "On the Reliability of Linear Regression and Pattern Recognition Feedforward Artificial Neural Networks in FPGAs," *IEEE Transactions on Nuclear Science*, vol. 65, no. 1, pp. 288–295, January 2018.

[6] F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, and P. Rech, "Selective Hardening for Neural Networks in FPGAs," *IEEE Transactions on Nuclear Science*, vol. 66, no. 1, pp. 216–222, January 2019.

[7] H.-B. Wang, Y.-S. Wang, J.-H. Xiao, S.-L. Wang, and T.-J. Liang, “Impact of Single-Event Upsets on Convolutional Neural Networks in Xilinx Zynq FPGAs,” *IEEE Transactions on Nuclear Science*, vol. 68, no. 4, pp. 394–401, April 2021.

[8] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, “Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications,” in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2017, pp. 84–95.

[9] S. Mittal, “A Survey on Modeling and Improving Reliability of DNN Algorithms and Accelerators,” *Journal of Systems Architecture*, vol. 104, p. 101689, March 2020.

[10] J. Gava, R. Reis, and L. Ost, “RAT: A Lightweight Architecture Independent System-Level Soft Error Mitigation Technique,” in *IEEE VLSI-SoC: Design Trends*, 2021, pp. 235–253.

[11] F. F. d. Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, “Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs,” *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, June 2019.

[12] M. G. Trindade, A. Coelho, C. Valadares, R. A. C. Viera, S. Rey, B. Cheymol, M. Baylac, R. Velazco, and R. P. Bastos, “Assessment of a Hardware-Implemented Machine Learning Technique Under Neutron Irradiation,” *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1441–1448, July 2019.

[13] R. M. Brewer, S. L. Moran, J. Cox, B. D. Sierawski, M. W. McCurdy, E. X. Zhang, S. S. Iyer, R. D. Schrimpf, M. L. Alles, and R. A. Reed, “The Impact of Proton-Induced Single Events on Image Classification in a Neuromorphic Computing Architecture,” *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 108–115, January 2020.

[14] M. G. Trindade, R. P. Bastos, R. Garibotti, L. Ost, M. Letiche, and J. Beaucour, “Assessment of Machine Learning Algorithms for Near-Sensor Computing under Radiation Soft Errors,” in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2020, pp. 494–497.

[15] L. M. Luza, D. Söderström, G. Tsiligiannis, H. Puchner, C. Cazzaniga, E. Sanchez, A. Bosio, and L. Dilillo, “Investigating the Impact of Radiation-Induced Soft Errors on the Reliability of Approximate Computing Systems,” in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2020, pp. 49–54.

[16] S. Blower, P. Rech, C. Cazzaniga, M. Kastriotou, and C. D. Frost, “Evaluating and Mitigating Neutrons Effects on COTS EdgeAI Accelerators,” *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1719–1726, June 2021.

[17] F. Libano, P. Rech, B. Neuman, J. Leavitt, M. Wirthlin, and J. Brunhaver, “How Reduced Data Precision and Degree of Parallelism Impact the Reliability of Convolutional Neural Networks on FPGAs,” *IEEE Transactions on Nuclear Science*, vol. 68, no. 5, pp. 865–872, May 2021.

[18] R. L. Rech Junior, S. Malde, C. Cazzaniga, M. Kastriotou, M. Letiche, C. Frost, and P. Rech, “High Energy and Thermal Neutron Sensitivity of Google Tensor Processing Units,” *IEEE Transactions on Nuclear Science*, vol. 69, no. 3, pp. 567–575, March 2022.

[19] R. Possamai Bastos, M. G. Trindade, R. Garibotti, J. Gava, R. Reis, and L. Ost, “Assessment of Tiny Machine-Learning Computing Systems Under Neutron-Induced Radiation Effects,” *IEEE Transactions on Nuclear Science*, vol. 69, no. 7, pp. 1683–1690, July 2022.

[20] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, January 2004.

[21] R. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies,” *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, September 2005.

[22] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, “Ares: A framework for quantifying the resilience of deep neural networks,” in *ACM/IEEE Design Automation Conference (DAC)*, 2018, pp. 97–102.

[25] G. Abich, J. Gava, R. Garibotti, R. Reis, and L. Ost, “Applying Lightweight Soft Error Mitigation Techniques to Embedded Mixed Precision Deep Neural Networks,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 11, pp. 4772–4782, November 2021.

[23] F. R. da Rosa, R. Garibotti, L. Ost, and R. Reis, “Using Machine Learning Techniques to Evaluate Multicore Soft Error Reliability,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2151–2164, June 2019.

[24] Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, “BinFI: An Efficient Fault Injector for Safety-Critical Machine Learning Systems,” in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2019, pp. 1–23.

[26] J.-D. Guerrero-Balaguera, R. L. Sierra, and M. S. Reorda, “Effective fault simulation of GPU’s permanent faults for reliability estimation of CNNs,” in *IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2022, pp. 106–111.

[27] A. Colucci, A. Steininger, and M. Shafique, “enpheeph: A Fault Injection Framework for Spiking and Compressed Deep Neural Networks,” 2022. [Online]. Available: https://arxiv.org/abs/2208.00328

[28] Z. Liu, Z. Deng, and X. Yang, “Using checksum to improve the reliability of embedded convolutional neural networks,” *Microelectronics Reliability*, vol. 136, p. 114666, September 2022.

[29] N. Narayanan, Z. Chen, B. Fang, G. Li, K. Pattabiraman, and N. Debardeleben, “Fault Injection for TensorFlow Applications,” *IEEE Transactions on Dependable and Secure Computing*, pp. 1–18, July 2022.

[30] S. Kundu, K. Basu, M. Sadi, T. Titirsha, S. Song, A. Das, and U. Guin, “Special Session: Reliability Analysis for AI/ML Hardware,” in *IEEE VLSI Test Symposium (VTS)*, 2021, pp. 211–220.

[31] STMicroelectronics, “STEVAL-STLKT01V1,” 2022. [Online]. Available: https://www.st.com/en/evaluation-tools/steval-stlkt01v1.html

[32] OctavoSystems, “OSD32MP15x System-in-Package,” 2022. [Online]. Available: https://octavosystems.com/octavo_products/osd32mp15x/

[33] Arm, “Auterion’s PX4,” 2022. [Online]. Available: https://www.arm.com/company/success-library/made-possible/px4

[34] J. Gava, V. Bandeira, F. Rosa, R. Garibotti, R. Reis, and L. Ost, “SOFIA: An automated framework for early soft error assessment, identification, and mitigation,” *Journal of Systems Architecture*, vol. 131, p. 102710, October 2022.

[35] G. Abich, R. Garibotti, V. Bandeira, F. da Rosa, J. Gava, F. Bortolon, G. Medeiros, F. G. Moraes, R. Reis, and L. Ost, “Evaluation of the soft error assessment consistency of a JIT-based virtual platform simulator,” *IET Computers & Digital Techniques*, vol. 15, no. 2, pp. 125–142, March 2021.

[36] L. Lai, N. Suda, and V. Chandra, “CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs,” 2018. [Online]. Available: https://arxiv.org/abs/1801.06601

[37] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” University Toronto, Tech. Rep., 2009.

[38] J. Xiao, G. Liu, K. Wang, and Y. Si, “Cow identification in free-stall barns based on an improved Mask R-CNN and an SVM,” *Computers and Electronics in Agriculture*, vol. 194, p. 106738, March 2022.

[39] STMicroelectronics, “STM32 Nucleo L476RG,” 2022. [Online]. Available: https://www.st.com/en/evaluation-tools/nucleo-l476rg.html

[40] ——, “STM32 Nucleo F767ZI,” 2022. [Online]. Available: https://www.st.com/en/evaluation-tools/nucleo-f767zi.html

[41] C. Slayman, “JEDEC Standards on Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors,” in *Soft Errors in Modern Electronic Systems*, 2011, pp. 55–76.

[42] JEDEC, “Global Standards for the Microelectronics Industry,” 2006. [Online]. Available: https://www.jedec.org

[43] G. Abich, R. Garibotti, R. Reis, and L. Ost, “The Impact of Soft Errors in Memory Units of Edge Devices Executing Convolutional Neural Networks,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 679–683, March 2022.

[44] G. Cellere, S. Gerardin, M. Bagatin, A. Paccagnella, A. Visconti, M. Bonanomi, S. Beltrami, P. Roche, G. Gasiot, R. H. Sorensen *et al.*, “Neutron-induced soft errors in advanced flash memories,” in *IEEE International Electron Devices Meeting (IEDM)*, 2008, pp. 357–360.